SUSE Linux Enterprise Server

11 SP4

www.suse.com

July 14, 2015

Security and Hardening Guide

Security and Hardening Guide

Contents

About This Guide

The SUSE Linux Enterprise Server Security and Hardening Guide deals with the particulars of installation and set up of a secure SUSE Linux Enterprise Server server and additional post-install processes required to further secure and harden that installation. Security and hardening elements and procedures are best applied to a server both during installation and post-installation and aim to improve the fitness of the system for the purposes demanded by its administrator. The understanding of this guide is to support the administrator with the security related choices and decisions that the administrator will have to make. The individual steps and procedures are to be seen as a proposal, not as something that ultimately needs to be done. In many cases, this guide will even force the reader to discuss the usefulness towards the objectives that the measures may provide - or not.

Obviously, the objective is to improve the security value of the system. Definitions about the meaning of the term security vary, but we want to settle on one that is both simple, abstract and therefore possibly true for most IT solutions:

A good system does what it is expected to do, and it does it well.

A secure system is a good system that does nothing else.

The part with “nothing else” is certainly within the focus of this guide. The Linux system is architected in such way that security policies are enforced. These policies are (fairly generic and incomplete list):

- DAC - Discretionary Access Control: File and directory permissions as we know them: `chmod`, `chown`
- privileged ports: TCP and UDP ports 0-1023 as well as raw sockets are only to the super user
- other privileged operations: The loading of kernel modules, configuration of network interfaces, exclusively all security relevant settings of the Linux kernel, are operations that can only be done by the root user, eg. the user with the numeric userID 0.

Attacking a system means to attempt to overcome (eg. circumvent or break) these privilege boundaries in a way that the administrator of the system or the programmer of the corresponding subsystem has not taken into account.

A “hardened” system raises the bar for the attacker to make the system do what “he” wants by reducing the area that the system exposes to the attacker (often called attack surface), and by mitigating the risk that exists for the system if a part of it fails to handle untrusted input safely, thereby allowing actions within the context of this part of the system that were not intended by the programmer.

Security is about decisions, and whenever security is in (apparent) opposition to function, these decisions become trade-offs. While it can be argued that all systems should be set up to be as securely as possible, some levels of security and hardening may very well be overkill in some cases. Each system's operational environment has its own security requirements derived from business drivers or regulatory compliance mandates (e.g. SOX, HIPAA, PCIDSS, etc.) and an effective business requirements analysis should be performed in order to determine the right level of security and hardening to be applied to a server or defined as part of a baseline server build.

As a final note before we begin: You may encounter individual requirements in regulatory compliance frameworks that may not make sense from a technical perspective, or they do not serve the purpose of improving security. It may be a productive attitude to simply implement what is required, but whenever there is a contradiction to security, an informed discussion in the documentation serves the overall purpose of your regulative compliance framework much more than blindly obeying the specifications. Please feel encouraged to dispute list items that you think are counterproductive.

1 Assumptions and Scope

While in most cases in this document reference will be made to a single server target or host, the scope can generally be applied to more than one machine. We generally assume that the security target can cover one or more systems running SUSE Linux Enterprise Server.

We explicitly do *not* make any assumtions about the hostility of the network that the systems are connected to, or the cooperative nature of the users that leverage the services provided by the systems.

In turn, this means that you partially define your context on your own when reading through this document. You will need to broaden the meaning of individual portions to adopt it to your environment. In some cases, such as the use case of a server that is exposed to the Internet, this document may even be insufficient or incomplete; however, it may still serve as a good starting point on your journey towards an increased level of confidence that your system will behave like you want it to.

About trust: Trust relationships exist among all systems that participate in networked transactions. Basically, the trust relationship between the persons that use the systems is transported across these systems. The chain that is formed by your trust relationships is only as strong as the weakest link. If we further assume that not all your problems are between keyboards and chairs, then it is up to the designer of the network of systems to watchguard the trust relationships. It is good practice to graphically visualize the trust relationships with the services in a schematic overview or map of your network. Generally, it is up to the owner of a resource to enforce the policies imposed on that resource; this would usually be the server that provides the resource. The client that opens a connection to request the resource can only be made responsible for the actions that it performs. This refers to the action of opening the connection to start with, but to nothing else as such.

The case of hostile users is special and unique: The Human Resources department may be able to solve some of your security problems in your computing environment at least as well as some technical measures can. Please make sure that the necessary regulations in your environment fit your needs, and that they back your intentions instead of obstructing them if you need to work around a missing support from your HR department (and your management).

Persons that have administrative privileges on a system are automatically considered trusted.

A Linux system - without any additional security frameworks such as SELinux - is a single level security system: From a security policy perspective there is only the superuser (root) and non-privileged users. System users are non-root userIDs that have access to files specific to their purpose. All system user identities are inaccessible for local or networked users!) The separation of (systems-) administrative duties is complicated by this simplicity. Some tools however help: Make use of sudo(8) for administrative tasks, but be aware that once the privilege boundary is crossed, a program running with root privileges does not enforce any file access policies for non-privileged users any more. vi(1) that runs as root can read and write to any file in the system.

Another tool to mitigate the risk of the abuse or accidential misuse of administrative privileges is Novell's Privileged User Manager product. More information is available here:

- http://www.novell.com/products/privilegedusermanager/index.html

Physical security of the server is another assumption made here, where the server is protected from theft and manipulation by unauthorized persons. A common sobering

thought amongst security professionals is the “ten-second Denial of Service” simply unplug the wires and reboot the server. Physical security must be insured and physical access must be controlled. Otherwise, all assumtions about at least the avaliability of these systems are void.

NOTE: Cryptography

The use of cryptography to protect the confidentiality of transactions with the services that your system provides is generally encouraged. The need to implement crypto-enhancements is strongly dependent on the operational environments of all participating systems. Please keep in mind that you need to verify all of the possible security benefits that cryptography can provide, for “all” of your services, and that these benefits are not delivered automatically just by turning on the “encrypt” option of your service (if you can enjoy the idyllic situation where encryption is available as a button to check):

Confidentiality
: Protection against reading the “content”of a transaction

Privacy
: Protection against knowing that a transaction exists, and some properties that it may have, such as size, identities of involved parties, their presence, ...

Integrity
: Protection against alteration of content. Be aware that cryptography does not automatically provide this kind of protection.

Authenticity
: Protection against identity fraud. Cryptogrphy that does not know about identities of participating entities cannot deliver this value. If an ecrypted data connection to your server can articulate integrity, then the server “may” provide authenticity of the content, but the cryptographical part cannot.

Keep in mind that encryption of data for confidentiality purposes can merely reduce the size of the data to protect from the actual size to the size of the key that is used to encrypt the data. This results in a key exchange problem for encrypted transactions, and in a key management problem for encrypted data storage. Since data is (typically, there are exceptions!) processed in clear, you need your vault unlocked while data within is being worked with.

The encryption of such data on the file system or block device layer helps against the theft of the system, but it doesn't help the confidentiality of the data while the system is running.

If you want to implement a consistent security policy covering multiple hosts on a network then organizational procedures *must* ensure that all those hosts can be trusted and are configured with compatible security configurations enforcing an organization wide security policy. Isolation of groups of systems that maintain data of the same trust domain can provide an adequate means of control; ultimately, the access controls to these systems, both for end users and for other systems, need to be carefully designed, configured, inspected and monitored.

IMPORTANT

Data can only be trusted to the degree that is assiciated with the domain where it comes from. If data leaves the domain in which security policies can be enforced, then this data should consequently be associated with the trust of the target domain. (This is the first facepalm paragraph of this guide. Do not stop fixing occurrences of failure and misconfiguration in your network - it pays off.)

For a review of industry best practices on security, the development of sound security processes, controls, development, reviews and audit practices and incident management, you can review a public RFC (request for comments). RFC2196 is the ongoing work of the world-wide community and individual security and process experts. You can review it online here: `http://www.faqs.org/rfcs/rfc2196.html`. An RFC is an open and living document that invites "comments" and review". Enhancements and improvements are welcomed; you will find instructions on where to send those suggestions within the document itself.

This guide provides initial guidance on how to set up and secure a SUSE Linux Enterprise Server installation but it is not intended to be the only information required for a system administrator to learn how to operate Linux securely. Assumptions are made within this guide that the reader has knowledge and understanding of operating security principles in general, and of Linux administrative commands and configuration options in particular. Upon reaching this section of the document, we believe that the willingness of the reader to learn can be safely assumed.

The guide contains “Parts”, “Chapters”, “Sections” and many examples. The “Parts” are a complete set of guidance and recommendations that can be used as a stand-alone

reference within a specific context. For example, Part 1 contains a reference to Common Criteria and SUSE Linux Enterprise Server. Part 2 contains more general system security and service protection schemes.

The “Chapters” will divide the books into logical parent topics within the “parts” and likewise the “sections” sub-divide even further.

Examples will exist throughout the whole guide, and many more can be found in referenced man pages or documentation.

2 Available Documentation

We provide HTML and PDF versions of our books in different languages. The following manuals for users and administrators are available for this product:

Deployment Guide (↑*Deployment Guide*)
: Shows how to install single or multiple systems and how to exploit the product inherent capabilities for a deployment infrastructure. Choose from various approaches, ranging from a local installation or a network installation server to a mass deployment using a remote-controlled, highly-customized, and automated installation technique.

Administration Guide (↑*Administration Guide*)
: Covers system administration tasks like maintaining, monitoring, and customizing an initially installed system.

Security Guide (↑*Security Guide*)
: Introduces basic concepts of system security, covering both local and network security aspects. Shows how to make use of the product inherent security software like AppArmor (which lets you specify per program which files the program may read, write, and execute), and the auditing system that reliably collects information about any security-relevant events.

Security and Hardening (page i)
: Deals with the particulars of installing and setting up a secure SUSE Linux Enterprise Server, and additional post-installation processes required to further secure and harden that installation. Supports the administrator with security-related choices and decisions.

System Analysis and Tuning Guide (↑*System Analysis and Tuning Guide*)
: An administrator's guide for problem detection, resolution and optimization. Find how to inspect and optimize your system by means of monitoring tools and how to efficiently manage resources. Also contains an overview of common problems and solutions, and of additional help and documentation resources.

Virtualization with Xen (↑*Virtualization with Xen*)
: Offers an introduction to virtualization technology of your product. It features an overview of the various fields of application and installation types of each of the platforms supported by SUSE Linux Enterprise Server as well as a short description of the installation procedure.

Virtualization with KVM for IBM System z (↑*Virtualization with KVM* for IBM System z)
: Offers an introduction to setting up and managing virtualization with KVM (Kernel-based Virtual Machine) on SUSE Linux Enterprise Server. Learn how to manage KVM with libvirt or QEMU. The guide also contains detailed information about requirements, limitations, and support status.

AutoYaST (↑*AutoYaST*)
: AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention, using an AutoYaST profile that contains installation and configuration data. The manual guides you through the basic steps of auto-installation: preparation, installation, and configuration.

Storage Administration Guide (↑Storage Administration Guide)
: Provides information about how to manage storage devices on a SUSE Linux Enterprise Server.

In addition to the comprehensive manuals, several quick start guides are available:

Installation Quick Start (↑*Installation Quick Start*)
: Lists the system requirements and guides you step-by-step through the installation of SUSE Linux Enterprise Server from DVD, or from an ISO image.

Linux Audit Quick Start
: Gives a short overview how to enable and configure the auditing system and how to execute key tasks such as setting up audit rules, generating reports, and analyzing the log files.

AppArmor Quick Start
: Helps you understand the main concepts behind AppArmor®.

Virtualization with Linux Containers (LXC) (↑*Virtualization with Linux Containers (LXC)*)

Gives a short introduction to LXC (a lightweight "virtualization" method) and shows how to set up an LXC host and LXC containers.

Find HTML versions of most product manuals in your installed system under `/usr/share/doc/manual` or in the help centers of your desktop. Find the latest documentation updates at `http://www.suse.com/doc` where you can download PDF or HTML versions of the manuals for your product.

3 Feedback

Several feedback channels are available:

Bugs and Enhancement Requests

For services and support options available for your product, refer to `http://www.suse.com/support/`.

To report bugs for a product component, log in to the Novell Customer Center from `http://www.suse.com/support/` and select *My Support* > *Service Request*.

User Comments

We want to hear your comments about and suggestions for this manual and the other documentation included with this product. Use the User Comments feature at the bottom of each page in the online documentation or go to `http://www.suse.com/doc/feedback.html` and enter your comments there.

Mail

For feedback on the documentation of this product, you can also send a mail to `doc-team@suse.de`. Make sure to include the document title, the product version, and the publication date of the documentation. To report errors or suggest enhancements, provide a concise description of the problem and refer to the respective section number and page (or URL).

4 Documentation Conventions

The following typographical conventions are used in this manual:

- `/etc/passwd`: directory names and filenames
- *`placeholder`*: replace *`placeholder`* with the actual value
- `PATH`: the environment variable PATH
- `ls`, `--help`: commands, options, and parameters
- `user`: users or groups
- Alt, Alt + F1: a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File*, *File* > *Save As*: menu items, buttons
- **#amd64 em64t ipf:** This paragraph is only relevant for the architectures `amd64`, `em64t`, and `ipf`. The arrows mark the beginning and the end of the text block. ◄

  #ipseries zseries: This paragraph is only relevant for the architectures `System z` and `ipseries`. The arrows mark the beginning and the end of the text block. ◄
- *Dancing Penguins* (Chapter *Penguins*, ↑Another Manual): This is a reference to a chapter in another manual.

Part I. SUSE Linux Enterprise Server and Common Criteria

1 Overview and Rationale

Common Criteria certificates have been published for every major release of the SUSE Linux Enterprise Server since version 8. SUSE Linux Enterprise Server 11 Service Pack 2 a Common Criteria v3.1 OSPP EAL4+ certificate in May 2013.

1.1 Common Criteria and this document

Common Criteria is the most known and most widely used methodology to evaluate and measure the security value of an IT product. The methodology aims to be independent, as an independent laboratory conducts the evaluation, which a certification body will certify afterwards. Security Functional Requirements (SFR) are summarized in so-called Protection Profiles (PP), which allows the comparison of security functions of different products if the definition of the Security Target (ST) (which typically makes use of a reference to the PP if one exists that fits the purpose of the product) and the Evaluation Assurance Levels are comparable.

As noted earlier, a clear definition of security in IT products is challenging. Security is to be considered a process that never ends, not a static condition that can be met or not. A Common Criteria certificate (below EA7) does not make a clear statement about error proneness of the system (while many of the flaws that exist specifically in operating systems are security-relevant), but it adds an important value to the product that cannot be described with the presence of technology alone: That someone has independently inspected the design of the system in such way that is corresponds to the claims that are

made, and that explicit care has been taken in producing and maintaining the product. In an environment where the product is reliably shielded from any kind of attacker, the only remaining attack vector is the vendor of the system.

The certificate states a degree of maturity of both the product with its security functions and the processes of the company that has designed, built and engineered the product, and that will maintain the product across its life cycle. As such, Common Criteria aims to be fairly holistic with its approach to take everything into account that is relevant for the security of an IT product.

The Evaluation Assurance Level (EAL) shall denote the degree of confidence that the product fulfills the described claims. The levels are from 1 through 7:

- EAL1: Functionally tested
- EAL2: Structurally tested
- EAL3: Methodically tested and checked
- EAL4: Methodically designed, tested and reviewed
- EAL5: Semiformally designed and tested
- EAL6: Semiformally verified design and tested
- EAL7: Formally verified design and tested

While EAL1 only provides basic assurance for products to meet security requirements, EAL2 to 4 are medium assurance levels. EAL5-EAL7 describe medium-to-high and high assurance; EAL4 is expected to be the highest level of assurance that a product can have if it has not been designed from the start to achieve a higher level of assurance.

Basically all commonly known General Purpose/Utility Computing operating systems have been awarded a Common Criteria certificate at EAL4. A "+" after the assurance level denotes an augmentation to the EAL, an addition that is useful for the articulation of security value, but formally not needed at the corresponding EAL.

The SUSE Linux Enterprise Server version 8 was the first Linux system to achieve EAL3+ (Augmentation: Basic Flaw Remidiation) in 2003; Version 9 of SLES was the first Linux based operating system to reach EAL4+ in 2004. More certifications and

re-certifications have followed targeting SLES9 and SLES10-SP1, until the SUSE Linux Enterprise Server version 11 Service Pack 2 was evaluated in 2011/2012.

The Common Criteria evaluations inspect a specific configuration of the product in a so-called “evaluated configuration”. The “Administrator's Guide” is a document that comes with a Common Criteria certified product and describes the individual steps that need to be taken to install and configure the product to a state like it was evaluated.

Very often, the evaluated configuration is used as a reference for the secure installation of the SUSE Linux Enterprise Server. It is however incorrect to understand the evaluated configuration as a hardened configuration: the removal of setuid-bits and the prescription of administrative procedures after installation is there to reach a specific configuration that is sane, but this process is clearly insufficient for a hardening claim.

Earlier versions of this document have contained a substantial part that links to Common Criteria evaluated configurations. For the purpose of clarity and to avoid confusion, this version drafted for the SUSE Linux Enterprise Server version 11 Service Pack 2 has these chapters removed.

Instead, this guide recommends the lecture the documentation that comes with the Common Criteria certificate to understand the Common Critiera evaluation of SUSE Linux Enterprise Server in general, the security functions that are in place within the operating system and how these security functions are relevant for the mitigation of threats. The High Level Design documentation encompasses the design specifics of the SUSE Linux Enterprise Server: Authentication mechanisms, access controls, audit subsystem and system logs, just to name a few of them. The accumulated knowledge contained in the documentation allows decision making for hardening purposes at an informed level - find it at

[http://www.suse.com/security/]

Apart from the valuable documentation that comes with the Common Criteria effort, the following manual pages may be of greater interest to the inclined reader:

“pam(8), pam(5)”

“apparmor(7)” and referred man pages

“syslog(8), syslogd(8), rsyslogd(8)” (if rsyslogd is installed)

“fstab(5), mount(8), losetup(8), cryptsetup(8)”

“haveged(8), random(4)”

“ssh(1), sshd(8), ssh_config(5), sshd_config(5), ssh-agent(1), ssh-add(1), ssh-keygen(1)”

“cron(1), crontab(5), at(1), atd(8)”

“mkinitrd(8), mkinitrd(5), init(8), inittab(5), init.d(7), runlevel(8)”

1.2 Generic Guiding Principles

The following guiding principles motivate much of the advice in this guide, and security processes in general, and should also influence any configuration decisions that are not explicitly covered.

Use Data Encryption Whenever Possible

Please refer to the "About... " section of this guide. In "Assumptions", the limitations of cryptography are briefly outlined.

Please be aware that cryptography is certainly useful, but only for the specific purposes that it is good for. It is not a generic recipe for better security in a system, its use may even impose additional risk on the system. Make informed decisions about the use of cryptography, and feel obliged to have a reason for your decisions, no matter if they are for or against cryptography. A false sense of security is normally more harmful than the weakness itself. SUSE Linux Enterprise Server supports encryption for generic network connections (the openssl command, stunnel), for remote login (openssh, man ssh(1)), for generic file encryption (gpg), for entire file systems at block layer (dm_crypt, cryptsetup) and for VPN (IPsec, openvpn).

Minimal Package Installation

Generally, an RPM software package consists of the package's metadata that is written to the RPM database upon installation, the package's files and directories and scripts that are being executed before and after installation and uninstallation.

If the package does NOT contain

1. setuid- or setgid bits on any of the installed files

2. group- or world-writeable files or directories

3. a service that is activated upon installation/activated by default.

then the said package generally does not impose any security risk to the system. Under the above condition, the package is merely a collection of files, and their use shall not be automatically assumed if you do not have any local users on your system. Since the installation of such packages does not have any influence on the security value of the system, the uninstallation of them shouldn't either.

However, a fairly simple reason to keep to a minimal set of packages in your installation is that something that is not present cannot get used. Binaries not installed cannot be executed.

A straight forward way of keeping to a minimal set of packages begins with the installation of the system. You can start the installation of your system by deselecting all packages and then select only those that you wish to use. As an example, the selection of the apache2-mod_perl package in yast2 would automatically cause all packages to be selected for installation that are needed for the apache package to operate. In many cases, dependencies have been artificially cut down to be able to handle the system's dependency tree more flexibly. You should be safe if you chose the minimal system, and build the dependency tree from there with your (leaf) package selection.

Service Isolation - Run Different Services on Separate Systems

Whenever possible, a server should be dedicated to serving exactly one service or application. This limits the number of other services that could be compromised in the event an attacker is able to successfully exploit a software flaw in one service (assuming that flaw allows access to others).

This point can lead to healthy and robust “dialog” on system sizing and even further to consolidation or virtualization. The intent with this guidance is to reduce the fault domain and risk where possible.

The use of AppArmor for services that are provided on a system is an effective means of containment. Please refer to the AppArmor documentation on your system to learn more. “man apparmor” is a good starting point.

The use of virtualization technology with kvm or with xen is supported with the SUSE Linux Enterprise Server version 11. While virtualization is generally designed for server consolidation purposes, its usefulness for service isolation is another good argument. Please be aware that the capability of the hypervisor to seperate virtual machines is not higher or stronger than the Linux kernel's capabil-

ity to seperate processes and their address spaces. The granularity at which virtualization technology tackles separation may however come with its benefits, being resource-hungry and somewhat clumsy on the other hand.

NOTE

Virtualization technology cannot match or subsitute the separation strength that is given by running services on different physical machines!

System fingerprinting and backups

In the case of the suspicion of an attack against the system, nothing can provide more comfort than

1. a backup

2. a fingerprint of your system to detect modifications

3. having done your homework.

Several tools exist on SUSE Linux Enterprise Server 11 which can be effectively used for the detection of unknown, but yet successful attacks. These tools come at the cost of relatively little configuration effort, but with the benefit of being able to actually “know” what has been changed in your system.

In particular, the use of `AIDE` is strongly encouraged. `AIDE`, when run for initialization, creates a hash database of all files in the system that are configured. This allows to verify the integrity of all catalogued files at a later time.

NOTE

You need to copy the hash database that AIDE creates to a place that is unaccessible for potential attackers. Otherwise, the attacker may modify the integrity database after planting a backdoor, thereby defeating the purpose of the integrity measurement.

NOTE

An attacker may have planted a backdoor in the kernel. This has an entire variety of consequences: Apart from being very hard to detect, the kernel based backdoor can effectively remove all traces of the system compromise to the degree that system alterations are basically invisible.

By consequence, an integrity check needs to be done from a rescue system (or any other system where an independent system runs, and the target system's file systems are mounted manually).

NOTE

Security is a lively process. Essentially, in this context, this means that the application of security updates invalidates the integrity database. “rpm -qlv packagename” lists the files that are contained in a package. Generally spoken, the RPM subsystem is very powerful with the data that it maintains, and that is accessible with the “--queryformat” commandline option. A differential update of integrity database with the changed files becomes more manageable with some fine-grained utilisation of RPM.

A fast and directly accessible backup adds distinct confidence about the integrity of your system and can substitute an integrity check such as described above with AIDE. It is important, though, that the backup mechanism/solution has adequate versioning support so that you can trace changes in the system. As an example: The installation times of packages (“rpm -q --queryformat='%{INSTALLTIME} %{NAME}\n' packagename” must correspond to the changed files in the backup's logs.

NOTE

Needless to say, you would need to verify every now and then that your backup actually works as intended. Make this an integral part of your security routine!

Part II. General System Security and Service Protection Methods

2 Introduction

In Part I, “SUSE Linux Enterprise Server and Common Criteria” (page 1) we covered the Common Criteria EAL 4+ certified installation and setup that was sponsored by IBM for a select subset of hardware. This “certified” build is a great first-stop for customers wishing to build a secure and hardened base system, yet might not address all of the services and software specifics that many customers would be interested in.

This next part will present more of a “generalist” view and give general recommendations and guidance for SUSE Linux Enterprise Server system security. Some topics may seem repeated here (from the previous part) yet the context is very different. More detail will be provided in some sections and certainly some more general examples for a greater number of services.

3

Linux Security in "General"

In this portion of the guide, instead of taking a *must*, *recommended* or *may* approach to the setup of a secured host, we will address security topics in a more "general" fashion. This is where you will get a rule-of-thumb or best-practice even a basic recommendation. The procedures and examples here should give you the ability to apply security enhancement techniques to a wider variety of server-based services and programs.

Even if some of these topics might have been covered in "explicit" terms earlier with no leniency towards implementation variation (in accordance with the EAL target evaluation) you may find the examples that follow to contain more detail or explanation. Some of the general topics will include:

- Physical Security – Protection of the server from environmental threats (people, places, things).
- Security Policies and Procedures – Server lifecycle management, disk/media reclamation, backup and archive security.
- Systems Monitoring – Procedures around event notification/management.
- Systems Automation – Mechanisms and/or procedures for automatic security measures. Heuristics, account control, security reporting and remediation, automated shutdown, etc.
- Systems Management – Methods to obtaining packages, verification and signing keys, patching procedures and recommendations.
- Securing Network – Addition programs, ports and service wrappers – iptables, tcp-wrappers, services.

- Remote Access – extra SSH information and key federation. CA integration.
- Common Services – mail, nfs and automount.
- Securing the Kernel and Init Process – parameters, inittab, runlevels and boot scripts.
- Access Control – user/groups/permissions.
- Password Security and Warnings – Proper setup of passwords, banners and `xinetd`.
- Miscellaneous Security – Assorted security settings and miscellaneous stuff.
- Resources – Web links, documentation and example references, HOWTOs and general information, product links.

The sections will again be organized by a topical hierarchy for continuity-sake. Refer to the main table of contents for easy reference.

NOTE: Typography

The following examples will show the `>` symbol to denote a "normal" user (without `root` privileges) and also the `#` will denote commands run as the "`root`" user.

3.1 Physical Security

Physical security should be one of the utmost concerns if not the one of the primary ones. Linux production servers should be in locked datacenters where only people with passed security checks have access. Depending on the environment and circumstances, you also may want to consider boot loader passwords. Some questions to consider: Who has direct physical access to the host? Of those that do, should they? Can the host be protected from tampering? Should it be? Etc.

The amount of physical security needed on a particular system depends on the situation, and can also vary widely by available funds.

3.1.1 System locks

Most data center server racks include a “locking” feature. Usually this will be a hasp/cylinder lock on the front of the rack that allows you to turn an included key to a locked or unlocked position – granting or denying entry. Cage locks can help prevent someone from tampering or stealing devices/media from the servers, or opening up the cases and directly manipulating/sabotaging the hardware. Preventing system reboots or the booting from alternate devices is also important (e.g. floppys, CD/DVDs/USB drives/etc.).

Some servers also have case locks. These locks can do different things according to the designs of the system vendor and construction. Many systems are designed to self-disable if attempts are made to open the system without unlocking. Others have device covers that will not let you plug in or unplug new keyboards or mice. While locks are sometimes a useful feature, they are usually lower quality and easily defeated by attackers with ill intent.

3.2 Locking down the BIOS

The BIOS (Basic Input/Output System) is the lowest level of software/firmware that dictates system configuration and low-level hardware. GRUB, LILO and other Linux boot loaders access the BIOS to determine how to boot the host. Other hardware types (POWER/System z) that run Linux also have low-level software/firmware. Typically the BIOS can be configured to help prevent attackers from being able to reboot the host and manipulate the system.

Most BIOS varieties allow the setting of a boot password. While this does not provide a high level of security (a BIOS can be reset, removed or modified – assuming case access), but it can be another deterrent.

Many BIOS capabilities have other various security settings – checking with the system vendor, the system documentation or examine the BIOS during a system boot.

IMPORTANT: Booting when a BIOS Password is Set

If a system host has been set up with a boot password, the host will not boot up unattended (e.g. a system reboot, power failure, etc.). This is the trade-off/risk.

3.3 Security via the Boot Loaders

Linux boot loaders, like GRUB (used by default in SUSE Linux Enterprise Server) and LILO (optional), can also have a boot passwords set. LILO has the password and restricted settings. The password setting requires a password at boot time, and the restricted setting requires a boot-time password only if you specify different boot options (such as single) at the LILO prompt.

GRUB also provides a “password” feature, so that only administrators can start the interactive operations (i.e. editing menu entries and entering the command-line interface). If a password is specified, GRUB will disallow any interactive control until you press the key P or E and enter a correct password.

You can refer to the GRUB and LILO man pages for examples.

It is very important to keep in mind that when setting these passwords they will obviously need to be remembered! Also, enabling these passwords might merely slow an intrusion, not necessarily prevent it. Again, someone could boot from a floppy, and mount your root partition. If you are using BIOS-level security as well as a boot loader, it is a good practice to disable the ability to boot from floppy or other devices in your computer's BIOS, and then also password-protecting the BIOS itself.

Also keep in mind that the boot loader config files will need to be protected by changing their mode to `600` (read/write for `root` only), or others will be able to read your passwords or hashes!

3.4 Verifying Security Action with seccheck

Verifying Security Action ItemsIt is highly recommended to have scripts in place which can verify that security actions or procedures have been run. Even the best systems administrators can make errors or forget something. If you have a small or large Linux installation or environment, you should consider the use of the seccheck scripts.

Seccheck is the SUSE Security Checker. It is a set of several shellscripts designed to check the local security of the system on a regular basis. There are three main scripts that are executed at different time intervals. They are `security-daily`, `security-weekly` and `security-monthly`. At installation these scripts all have

schedule entries that get placed in cron that determine when they run. Although cron scheduling is the default behavior, this can be controlled via configuration settings (see next section). The daily script runs at midnight, and if changes are detected since the last run (the night before), an e-mail noting the differences will be sent. The weekly script runs every Monday at 1:00am, and only if changes to the last run (the week before) are found, a mail with the differences will be sent. The monthly script runs every on every 1st of the month and sends the full last daily and weekly report via e-mail.

3.4.1 Seccheck Configuration

Please note that you can change the receiver of the seccheck mails from root to anyone else if you add an entry like this one to `/etc/sysconfig/seccheck`:

```
SECCHK_USER="firewall" # exchange firewall is an admin user's account name
```

Please also note that the `START_SECCHK` variable from `/etc/sysconfig/seccheck` controls whether the security check will be run from cron. (It's ignored, if you call `security-control` manually.)

The following daily checks are done:

`/etc/passwd` check	length/number/contents of fields, accounts with same uid accounts with uid/gid of 0 or 1 beside root and bin
`/etc/shadow` check	length/number/contents of fields, accounts with no password
`/etc/group` check	length/number/contents of fields
user root checks	secure umask and `PATH`
`/etc/ftpusers`	checks if important system users are put there
`/etc/aliases`	checks for mail aliases which execute programs
`.rhosts` check	checks if users' `.rhosts` file contain + signs

homedirectory	checks if homedirectories are writable or owned by someone else
dot-files check	checks many dot-files in the homedirectories if they are writable or owned by someone else
mailbox check	checks if user mailboxes are owned by user and unreadable
NFS export check	exports should not be exported globaly
NFS import check	NFS mounts should have the `nosuid` option set
promisc check	checks if network cards are in promiscious mode
list modules	just lists loaded modules
list sockets	just lists open ports
Weekly Checks are as follows:	
password check	runs john to crack the password file, user will get an e-mail notice to change his password asap
RPM md5 check	checks for changed files via RPM's md5 checksum feature
suid/sgid check	lists all suid and sgid files
exec group write	lists all executables which are group/world writeable
writable check	lists all files which are world writable (incl. Above)

device check	lists all devices

IMPORTANT

For the weekly seccheck password check to work the “john the ripper” package needs to be installed yet could be considered a security risk in itself. It's important to understand the operation of john. Normally it will be used only to send notice of weak passwords to users as an e-mail as part of the weekly seccheck cron job. Since “john” would need to be installed separately, it is recommended to weigh the benefits and risks before doing so.

Additional monthly checks are also run, however the key difference is mainly that the monthly file is not a “diff” like the daily/weekly ones but the full reports in one file.

3.5 Retiring Linux Servers with Sensitive Data

Security policies usually contain some procedures for the treatment of storage media that is going to be retired or disposed of. Disk and media “wipe” procedures are frequently prescribed as is complete destruction of the media. You can find several free tools on the Internet. A search of “dod disk wipe utility” - will yield several variants. To retire servers with sensitive data, it is important to ensure that data cannot be recovered from the hard disks. To ensure that all traces of data are removed, a wipe utility can be used. Some of these tools can even be operated from a floppy disk (bootable) and remove data according to the U.S. Department of Defense (DoD) standards. Note that many government agencies specify their own standards for data security. Some standards are stronger than others, yet may require more time to implement. “DiskSanitizer” is just one of many that you can find. This utility is shareware and available to download at http://freshmeat.net/projects/disksanitizer.

3.6 Backups

If your system gets compromised, your backups become invaluable. But also in cases like bugs, accidents etc. backups can be used to compare you current system against your backed-up system. For production systems it is very important to take some Back-

ups offsite for cases like disasters (e.g. offsite storage of tapes/recordable media, or offsite initiated).

For legal reasons, some firms and organizations must be careful about backing up too much information and holding it too long. If your environment has a policy regarding the destruction of old paper files, you might have to extend this policy to Linux backup tapes as well.

3.7 Disk Partitions

Servers should have separate partitions for at least `/`, `/boot`, `/usr`, `/var`, `/tmp`, and `/home`. You don't want that e.g. logging and temporary space under `/var` and `/tmp` fill up the root partition. Third party applications should be on separate file systems as well, e.g. under `/opt`.

Review Part I, "SUSE Linux Enterprise Server and Common Criteria" (page 1). To repeat: It is important to understand the need to separate the partitions that could impact a running system (for example, log files filling up `/var/log` are a good reason to separate `/var` from the `/` partition). Another thing to keep in mind is that you will likely need to leverage LVM or another volume manager or at the very least the "extended" partition type to allow for the primary partition limitations (4 partitions).

Another capability in SUSE Linux Enterprise Server is encrypting a partition or even a single folder or file as a container. Please refer to Chapter 11, *Encrypting Partitions and Files* (↑*Security Guide*) for details.

3.8 Firewall (iptables)

We won't cover iptables in explicit detail in this guide. Most companies use hardware based firewalls to protect their servers in a production network, which is strongly recommended for secure environments. However, SUSE Linux Enterprise Server includes the SUSE-Firewall which is a wrapper for iptables and is enabled by default as a simple and layered protection.

If you are also interested in Linux stateful firewalls using iptables, there are several HOWTOs on the Internet. See the Appendix for resources. For lots of iptables tutorials and examples, see `http://www.linuxguruz.com/iptables/`. An overview is also available at Section "Packet Filtering with iptables" (Chapter 15, *Masquerading and Firewalls*, ↑*Security Guide*).

3.9 Security Features in the Kernel

The following list shows tunable kernel parameters you can use to secure your Linux server against attacks. Some of them are defaults already within the SLE distributions. To check the current status of any of these settings, you can query the kernel (`/proc/sys/...` contents). For example, the Source Routing setting is located in the `/proc/sys/net/ipv4/conf/all/accept_source_route` file. Simply cat the contents of a file to see how the current running kernel is setup.

For each tunable kernel parameter shown, the change to the entry that needs to be affected can be modified or added to the `/etc/sysctl.conf` configuration file to make the change persistent after a reboots.

You can get a list of current kernel settings by using the command:

```
sysctl -a
```

It is even a very good idea to store the output of the kernel settings (for comparison or reference) by redirecting the output of the sysctl command to a file, e.g.

```
sysctl -A > /root/sysctl.settings.store
```

Because SUSE Linux Enterprise Server includes, by default, security-focused kernel tuning parameters, you will find the existing `/etc/sysctl.conf` file to be sparsely populated. You may choose to use the above mentioned “catalog” method of storing the complete gamut of kernel settings and then pick-and-choose those parameters you want to be reset at reboot. You can place these in the `/etc/sysctl.conf` file or they can be inserted immediately (into the running kernel) by running the command `sysctl -p` or they will be picked up upon a reboot.

Many third party applications like Oracle, SAP, DB2, Websphere, etc. recommend changing kernel parameters to ensure high performance for I/O or CPU processing. Having a full list of current settings can be helpful for reference.

3.9.1 Enable TCP SYN Cookie Protection (default in SUSE Linux Enterprise Server11)

A “SYN Attack” is a denial of service attack that consumes all the resources on a machine. Any server that is connected to a network is potentially subject to this attack. To

enable TCP SYN Cookie Protection, edit the `/etc/sysctl.conf` file and ensure the following line and value exists:

```
net.ipv4.tcp_syncookies = 1
```

3.9.2 Disable IP Source Routing (default in SUSE Linux Enterprise Server11)

Source Routing is used to specify a path or route through the network from source to destination. This feature can be used by network people for diagnosing problems. However, if an intruder was able to send a source routed packet into the network, then he could intercept the replies and your server might not know that it's not communicating with a trusted server.

```
net.ipv4.conf.all.accept_source_route = 0
```

3.9.3 Disable ICMP Redirect Acceptance

ICMP redirects are used by routers to tell the server that there is a better path to other networks than the one chosen by the server. However, an intruder could potentially use ICMP redirect packets to alter the hosts's routing table by causing traffic to use a path you didn't intend. To disable ICMP Redirect Acceptance, edit the `/etc/sysctl.conf` file and add the following line:

```
net.ipv4.conf.all.accept_redirects = 0
```

3.9.4 Enable IP Spoofing Protection (default in SUSE Linux Enterprise Server11)

IP spoofing is a technique where an intruder sends out packets which claim to be from another host by manipulating the source address. IP spoofing is very often used for denial of service attacks. For more information on IP Spoofing, I recommend the article *IP Spoofing: Understanding the basics*.

```
net.ipv4.conf.all.rp_filter = 1
```

3.9.5 Enable Ignoring to ICMP Requests

If you want or need Linux to ignore `ping` requests, edit the `/etc/sysctl.conf` file and add the following line:

```
net.ipv4.icmp_echo_ignore_all = 1
```

This cannot be done in many environments, as even some monitoring systems use a rudimentary ICMP (ping) to determine the health of the device on the network (or at least its ability to respond).

3.9.6 Enable Ignoring Broadcasts Request (default in SUSE Linux Enterprise Server11)

If you want or need Linux to ignore broadcast requests...

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

3.9.7 Enable Bad Error Message Protection (default in SUSE Linux Enterprise Server11)

To alert you about bad error messages in the network...

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

3.9.8 Enable Logging of Spoofed Packets, Source Routed Packets, Redirect Packets

To turn on logging for Spoofed Packets, Source Routed Packets, and Redirect Packets, edit the `/etc/sysctl.conf` file and add the following line:

```
net.ipv4.conf.all.log_martians = 1
```

NOTE

ue to the way SUSE Linux Enterprise Server is setup (with syslog) for network event tracking, keep in mind that this may cause a large amount of messages to be logged.

3.9.9 Virtual Address Space Randomization

Starting with the 2.6.x kernel releases Linux now uses address-space randomization technique to mitigate buffer overflows. For more information, see

- http://searchopensource.techtarget.com/tip/1,289483,sid39_gci1144658,00.html
- http://lwn.net/Articles/121845/

3.10 AppArmor

Included with SUSE Linux Enterprise Server, AppArmor is an application security tool designed to provide an easy-to-use security framework for your applications. AppArmor proactively protects the operating system and applications from external or internal threats, even zero-day attacks, by enforcing good behavior and preventing even unknown application flaws from being exploited. AppArmor security policies, called "profiles", completely define what system resources individual applications can access, and with what privileges. A number of default profiles are included with AppArmor, and using a combination of advanced static analysis and learning-based tools, AppArmor profiles for even very complex applications can be deployed successfully in a matter of hours.

AppArmor consists of:

- a kernel module, shipped with the SUSE Linux kernel, which enforces the security profiles.
- a collection of RPMs, also shipped with SUSE Linux, that provide:
    - a set of AppArmor profiles for numerous programs that ship with SUSE Linux Enterprise Server

- tools to create and manage new and existing AppArmor profiles
- a YaST user interface to manage reports and notification of security events
- documentation about the AppArmor tools

It is best to reboot a system after completing installation, so that AppArmor can confine all system daemons.

The AppArmor quick-start and administrative guides are available online here:

http://www.novell.com/documentation/apparmor/

For additional details and step-by-step instructions on the usage and configuration of AppArmor you can also refer to Part “Confining Privileges with AppArmor” (↑*Security Guide*).

3.11 SELinux

SELinux is an advanced technology for securing Linux systems. Included with “basic enablement” in SUSE Linux Enterprise Server 11, and included with some other distributions by default. Hardening Linux using SELinux technology, on its own, warrants its own security HOWTO and is out of scope for this guide. The *SELinux: NSA's Open Source Security Enhanced Linux* is very good in regards to SELinux setup and usage. As part of the “basic enablement”, SELinux will not be officially supported, but packages have now been added to SUSE Linux Enterprise Server 11 to enable its usage with minimal effort. While AppArmor is much easier to use and has a similar feature set, knowing both will most certainly be beneficial.

3.12 FTP, telnet, and rlogin (rsh)

The programs/protocols of FTP, `telnet`, and `rlogin` (rsh) are normally vulnerable to eavesdropping, which is one of the main reasons why secure alternatives such as `ssh`, `scp` or `sftp` should be used instead. It is highly recommended not to run the insecure services. Due to the high risk, this guide does not cover these services (other than vsftp). It would also be a good idea (and part of our guidance, see next section) not to have FTP and Telnet server RPMs installed on the system. Note that the EAL

4+ evaluation had vsftp installed. The “vs” stands for “very secure” - which is a differentiator here when compared to normal ftp.

3.13 Removing Unnecessary Software Packages (RPMs)

A very important step in securing a Linux system is to determine the primary function(s) or role(s) of the Linux server. Strive for a deterministic and specific view of what is installed and running on the system. Otherwise it can be difficult to understand what needs to be secured and hence securing these Linux systems proactively might prove ineffective. Therefore, it is very critical to look at the default list of software packages and potentially remove any unneeded package(s) or packages that don't comply with your defined security policies.

Doing this will result in a smaller number of packages that may require updates and will certainly simplify maintenance efforts if/when security alerts and/or patches are released. Refer to the note below regarding SUSE's efforts around JeOS (Just Enough OS). An example could be, Apache or Samba installed on your system - if you don't use them, remove them. Also, it is generally recommended and a best practice not to have development packages, desktop software packages (e.g. X Server) etc. installed on production servers.

IMPORTANT

Many 3rd Party vendors like Oracle and IBM have a requirement for both the desktop environment and the development libraries to run some of their installers. Many organizations will create a silent installation (response file) in a dev lab, so this isn't an impact to production security.

Also, other packages like FTP and Telnet daemons should not be installed as well unless there is a justified business reason for it (again, `ssh`, `scp` or `sftp` should be used as replacements).

One of the first action items should be to create a Linux image that *only* contains RPMs needed by the system and applications, and those needed for maintenance and/or troubleshooting purposes. A good approach is to start with a minimum list of RPMs and then add packages as needed. It may be time-consuming but worth the efforts.

NOTE: Just Enough Operating System (JeOS)

To this end, shortly after the release of SUSE Linux Enterprise Server 10 SP2, SUSE developed a program called the SUSE Appliance Program. Included with this program is component called “JeOS” (pronounced “juice”) which stands for “Just Enough Operating System”. JeOS has a very small footprint and can be built to fit the very specific needs of a system developer. Main uses of JeOS will be for hardware/software appliance or virtual machine development. Key benefits of JeOS are efficiency, higher performance, increased security and simplified management.

To generate a list of all installed RPMs - use the following command:

```
rpm -qa
```

To retrieve details about a particular RPM (from the RPM itself), run:

```
rpm -qi package_name
```

To check for and report potential conflicts and dependencies when deleting an RPM, run:

```
rpm -e --test package_name
```

This can be very useful, as running the removal command without a “test” can often yield a mass of complaints and require manual recursive dependency hunting.

For information on performing AutoYaST installations and how to build an image, see *AutoYaST* (↑*AutoYaST*).

3.14 Patching Linux Systems

Building an infrastructure for the purpose of patch management is another very important part of a proactive and secure production Linux environment.

It is recommended to have a written security policy and procedure to handle Linux security updates and issues. For example, a security policy should detail the time frame for assessment, testing, and roll out of patches. Network related security vulnerabilities should get the highest priority and should be addressed immediately within a short time frame. The assessment phase should occur within a testing lab, and initial roll out should occur on development systems first

A separate security log should detail what Linux security notices have been received, when patches have been researched and assessed, when patches have been applied etc.

At this time SUSE releases their patches in three categories, security, recommended and optional. There are a few options that can be used to keep systems patched, up to date and secure. Each individual system can register and then retrieve updates via the SUSE Update website using the included YaST tool – YaST Online Update. SUSE has also created the Subscription Management Tool (SMT) an efficient way to maintain a local repository of available/released patches/updates/fixes that systems can then pull from (reducing Internet traffic). SUSE also offers SUSE Manager for the maintenance, patching, reporting and centralized management of Linux systems, not just SUSE, but other distributions as well.

3.14.1 YaST Online Update

On a per-server basis, installation of important updates and improvements is possible using the YaST Online Update tool. Current updates for the SUSE Linux Enterprise family are available from the product specific update catalogs containing patches. Installation of updates and improvements is accomplished using YaST and selecting *Online Update* in the *Software* Group. All new patches (except the optional ones) that are currently available for your system will already be marked for installation. Clicking *Accept* will then automatically install these patches.

After installation is complete, click the *Finish* button. The system will be patched, current and up-to-date.

3.14.2 Automatic Online Update

YaST also offers the possibility to set up an automatic update. Select *Software* > *Automatic Online Update*. Configure a Daily or a Weekly update. Some patches, such as kernel updates, require user interaction, which would cause the automatic update procedure to stop. Check *Skip Interactive Patches* for the update procedure to proceed automatically.

In this case, run a manual Online Update from time to install patches that require interaction.

When *Only Download Patches* is checked, the patches are downloaded at the specified time but not installed. They must be installed manually. The patches are downloaded to the rug cache directory, `/var/cache/zmd/web`, by default. Use the command `rug get-prefs cache-directory` to get the current rug cache directory

3.14.3 Subscription Management Tool - SMT

The Subscription Management Toolfor SUSE Linux Enterprise goes one step further than the Online Update process by establishing a proxy system with repository and registration targets. This helps customers centrally manage software updates within the firewall on a per-system basis, while maintaining their corporate security policies and regulatory compliance.

The downloadable SMT (http://download.novell.com/index.jsp?product_id=&search=Search&families=&date_range=&keywords=subscription+management+tool&sort_by=&results_per_page=&x=20&y=10) SMT is integrated with Novell Customer Center (http://www.novell.com/customercenter/) and provides a repository and registration target that is synchronized with it. This can be very helpful in tracking entitlements in large deployments. The SMT maintains all the capabilities of Novell Customer Center, while allowing a more secure centralized deployment. It is included with every SUSE Linux Enterprise subscription and is therefore fully supported.

The SMT provides an alternative to the default configuration, which requires opening the firewall to outbound connections for each device to receive updates. That requirement often violates corporate security policies and can be seen as a threat to regulatory compliance by some organizations. Through its integration with Novell Customer Center, the SMT ensures that each device can receive its appropriate updates without the need to open the firewall, and without any redundant bandwidth requirements.

The SMT also enables customers to locally track their SUSE Linux Enterprise devices (i.e., servers, desktops, or Point of Service terminals) throughout their enterprise. Now they can easily determine how many entitlements are in need of renewal at the end of a billing cycle without having to physically walk through the data center to manually update spreadsheets.

The SMT informs the SUSE Linux Enterprise devices of any available software updates. Each device then obtains the required software updates from the SMT. The introduction of the SMT improves the interaction among SUSE Linux Enterprise devices within the network and simplifies how they receive their system updates. The SMT enables an infrastructure for several hundred SUSE Linux Enterprise devices per instance of each installation (depending on the specific utilization profile). This offers more accurate and efficient server tracking.

In a nutshell, the Subscription Management Tool for SUSE Linux Enterprise provides customers with:

- Assurance of firewall and regulatory compliance
- Reduced bandwidth usage during software updates
- Full support under active subscription from SUSE
- Maintenance of existing customer interface with Novell Customer Center
- Accurate server entitlement tracking and effective measurement of subscription usage
- Automated process to easily tally entitlement totals (no more spreadsheets!)
- Simple installation process that automatically synchronizes server entitlement with Novell Customer Center

3.15 Securing the Network - Open Network Ports Detection

Securing a server requires that you know what it is serving; what services are running. Default server installations may have services running that aren't self apparent and open network ports that they are using. So, one of the most important tasks is to detect and close network ports that are not needed. To get a list of listening network ports (TCP and UDP sockets), you can use the `netstat` service run the following command:

```
netstat -tulp
```

Be aware that `netstat` output can be wider than a default terminal screen. If the screen is too narrow, the options described above will likely cause the output to wrap and be less legible.

Below is an example of output from the above command:

```
Proto Recv-Q Send-Q Local            Foreign         State   PID/Program
                    Address          Address                 name
tcp   0      0      *:auth           *.*             LISTEN  2328/xinetd
tcp   0      0      local[...].:smtp *.*             LISTEN  2360/
sendmail:acce
tcp   0      0      *:ssh            *.*             LISTEN  2317/sshd
```

From the output above you can see that three tcp-based services are running and listening: xinetd, sendmail, and sshd. Sendmail should not be configured to listen for incoming network connections unless the server running it is a designated as a mail or relay server. Running a port scan from another server will be able to confirm that, but make sure to obtain proper permissions to scan/probe a machine on a production network.

IMPORTANT

Some organizations consider port scans without permission a security offense.

Using the nmap tool, just such a probe/scan can be run:

```
jupiter:~ # nmap -sTU remote_host
Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-12-10 22:51 CST
Interesting ports on venus (192.168.2.101):
(The 3131 ports scanned but not shown below are in state: closed)
PORT     STATE         SERVICE
22/tcp   open          ssh
113/tcp  open          auth

Nmap run completed -- 1 IP address (1 host up) scanned in 221.669 seconds
```

Note that running the `nmap` command can take quite a while (in this example almost 4 minutes) depending on the options used. If you remove the UDP port scan (leave out the `-U` option), then `nmap` will finish the port scan nearly immediately. Running it on a local machine will also make it complete quickly. The results of `nmap` can vary widely and might not show all listening network sockets depending on the status of the SuSEFirewall2 (or other) and if it has been setup to block any ports.

From the sample run above, you see that the `xinetd` daemon is listening on port auth (port 113) for `IDENT` (for more information on this service, see Section 3.17, “`xinetd` Services - Disabling” (page 37)). You can also see that sendmail is not listening for remote incoming network connections.

Another method (safer) to list all of the TCP and UDP sockets to which programs are listening (on a host) is to use the lsof command – which lists open files :

```
jupiter:~ # lsof -i -n | egrep 'COMMAND|LISTEN|UDP'
COMMAND  PID   USER   FD   TYPE    DEVICE SIZE/OFF NODE NAME
sshd     2317  root   3u   IPv6    6579   0t0      TCP *:ssh (LISTEN)
xinetd   2328  root   5u   IPv4    6698   0t0      TCP *:auth (LISTEN)
sendmail 2360  root   3u   IPv4    6729   0t0      TCP 127.0.0.1:smtp
 (LISTEN)
```

3.16 Disabling Runlevel Services

One of the most important tasks is the removal of any network services from a systems start-up (init) process that are not needed. On SUSE systems you can list all services which are started at boot time using the following command:

```
chkconfig -l |grep on
```

This command will list any service that has been enabled to start at any of the 7 init levels. 0 through 6 and also S (single user). Notice there may be quite a few services enabled on a given host, but many runlevel services (Stand-Alone Services) might not be for network related services. There are some services that are only run during the boot process. Make sure NOT to disable any runlevel services that are needed by the system to run smoothly. For example, many of these type of services will begin with a prefix of `boot`.

Some examples of Runlevel System Services which you may or may not want to enable (check YaST *System Services (Runlevels)* module for a complete list):

Table 3.1: *init Services and their Descriptions - Examples*

Service Name	Description of Service
syslog	important for syslog services
netfs	needed only if there are NFS shares that should be mounted at boot time
network	important for starting network interfaces (e.g. eth0, eth1, bonding,...)
random	used for the system entropy pool
atd	needed if the at(1) service is used instead of cron
apmd	Advanced Power Management (APM) daemon is used for laptops and some desktops
isdn	needed if ISDN is being used

Service Name	Description of Service
iptables	needed if Netfilter (iptables) Firewall is being used
ip6tables	needed if ip6tables Firewall is being used
pcmcia	not needed on servers - needed for laptops
irqbalance	1. important for distributing interrupts across all CPUs
sendmail	needed if Sendmail is used - Procmail should be used which is more secure
autofs	needed if automounter is used - production applications should not be dependent on automounter
sshd	important for logins via SSH
portmap	needed if e.g. NFS is being used
nfslock	needed if NFS shares are mounted
nfs	1. needed if server runs the NFS server
mdmonitor	needed only if software RAID is being used
crond	important for running cron jobs
xinetd	needed if xinetd services are being used, see /etc/xinetd.d/ for list of services

Service Name	Description of Service
cups	needed if CUPS is used for the printing system
rhnsd	needed if server should connect to RHN to check for software updates etc.
sysstat	needed to reset system statistics logs
audit	needed only if Linux Audit Subsystem (LAuS) should run for collecting system call audit records
psacct	needed only if kernel process accounting information is needed
smartd	important for monitoring disk problems if hard disks support SMART technology
netdump	1. important if kernel oops data and memory dumps should be sent to a Netdump server for server crashes

The init (start/stop) scripts of all the runlevel services are found in the `/etc/init.d` directory. For example, if you don't know what the atd service does, go to `/etc/init.d` and open the file `atd`. And in the script look for lines that start programs. In the `atd` script the `startproc $ATD_BIN $ATD_ARGS` line starts the “process” `atd`. Now having the name of the program that is started by this service, you can check the online pages of `atd` by running `man atd`. This will help you to find out more about a system service.

To permanently disable e.g. the runlevel service nfs, run:

```
chkconfig nfs off
```

To immediately disable the runlevel service nfs, execute:

```
/etc/init.d/nfs stop
```

or just: `rcnfs stop`

3.17 `xinetd` Services - Disabling

The `xinetd` daemon is a replacement for inetd, the Internet services daemon. It monitors the ports for all network services configured in `/etc/xinetd.d`, and starts the services in response to incoming connections. To check if `xinetd` is enabled and running, execute:

```
jupiter:~ # chkconfig --list xinetd
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

To check the current status of the `xinetd` service, execute:

```
jupiter:~ # /etc/init.d/xinetd status
Checking for service xinetd: xinetd (pid 2619) is running..
```

or just: `rcxinetd status`

If `xinetd` is active, it is very important to see which services are active and being controlled by `xinetd`. The following command will list all services configured in `/etc/xinetd.d` and whether `xinetd` monitors the ports for these services:

```
jupiter:~ # chkconfig --list | awk '/xinetd based services/,/""/'
xinetd based services:

  chargen:     off
  chargen-udp: off
  cups-lpd:    off
  cvs:         off
  daytime:     off
  daytime-udp: off
  discard:     off
  discard-udp: off
  echo:        off
  echo-udp:    off
  netstat:     off
  rsync:       off
  sane-port:   off
  servers:     off
  services:    off
  svnserve:    off
  swat:        off
  systat:      off
  tftp:        on
  time:        off
  time-udp:    off
  vnc:         off
```

To get a list of only active services for which xinetd monitors the ports, you could run (where the -v option of grep does an inverse-match) :

```
jupiter:~ # chkconfig --list | awk '/xinetd based services/,/""/' | grep -v
 off
xinetd based services:

  tftp:        on
```

In the above example you can see that the `telnet-server` package is not installed on the system. If the Telnet Server package `telnet-server` would be installed, it would show up on the list whether it's active or not. Here is an example how to disable a service. Assuming the `tftp` service is active, run the following commands to disable it and to see how the telnet service entries are being updated:

```
jupiter:~ # chkconfig -l tftp
xinetd based services:

  tftp:        on
jupiter:~ # cat /etc/xinetd.d/tftp | grep disable
      disable = no
jupiter:~ # chkconfig tftp off
jupiter:~ # chkconfig --list tftp
xinetd based services:

  tftp:        off
jupiter:~ # cat /etc/xinetd.d/tftp | grep disable
      disable = yes
```

For the TFTP service it would be better to remove the package from the system since removal is always safer than just disabling (when possible):

```
rpm -e tftp
```

3.17.1 Inventory `xinetd` services

It is important to investigate *all* active `xinetd` services and to disable them (or remove their packages) if they are not needed. To find out what a service does, here is a viable approach. Using the `tftp` service as an example and assuming its function is unknown and it is listed as an active service. Execute the following commands:

```
jupiter:~ # grep " server" /etc/xinetd.d/tftp
  server = /usr/sbin/in.tftpd
  server_args = -s /tftpboot
```

To read the manual:

```
jupiter:~ # man in.tftpd
TFTPD(8)               System Manager's Manual              TFTPD(8)
```

```
NAME
       tftpd - IPv4 Trivial File Transfer Protocol server

SYNOPSIS
       in.tftpd [options...]  directory...

DESCRIPTION
       tftpd  is  a  server  for the Trivial File Transfer Protocol.
       The TFTP protocol is extensively used to support remote boot-
       ing  of  diskless devices.  The server is normally started by
       inetd, but can also run stand-alone.
[...]
```

To determine what package supplies the in.tftpd binary:

```
jupiter:~ # rpm -qf /usr/sbin/in.tftpd
tftp-0.48-101.16
```

To get a description of TFTP and its usage, etc:

```
jupiter:~ # rpm -qi tftp-0.48-101.16| awk '/Description/,/""/'
Description :
The Trivial File Transfer Protocol (TFTP) is normally used only for
booting diskless workstations and for getting or saving network
component configuration files.
```

To get a list of what files are installed via the TFTP package (RPM), execute the rpm command with the following options:

```
jupiter:~ # rpm -ql tftp-0.48-101.16
/etc/xinetd.d/tftp
/usr/bin/tftp
/usr/sbin/in.tftpd
/usr/share/doc/packages/tftp
/usr/share/doc/packages/tftp/README
/usr/share/doc/packages/tftp/README.security
/usr/share/doc/packages/tftp/sample.rules
/usr/share/man/man1/tftp.1.gz
/usr/share/man/man8/in.tftpd.8.gz
/usr/share/man/man8/tftpd.8.gz
```

This example described what could be done to find out information about services (specifically ones started by `xinetd`) even if an online manual didn't exist for the binary `in.ftpd`. This example yielded a man page – but you may not always find one. The RPM commands in the example are very commonly used for a variety of reasons. It is also possible to use the YaST software management interface to retrieve all of the resultant information – however having a knowledge of RPM command syntax can save quite a bit of time. Again to disable the TFTP service, execute the following command:

```
chkconfig tftp off
```

The `xinetd` daemon is quite flexible and has many features. Here are just a few functionalities of `xinetd`:

- Access control for TCP, UDP, and RPC services.
- Access limitations based on time.
- Provides mechanisms to prevent DoS attacks.

For more specific information on Xinetd, review the documentation and usage examples at the xinetd website: `http://www.xinetd.org/` and also for some xinetd usage tutorials: `http://www.macsecurity.org/resources/xinetd/tutorial.shtml`

3.18 Reviewing Inittab and Boot Scripts

The inittab file located in `/etc/inittab` contains a running "tab" (table) or list of which processes are to be started at bootup during normal system operation. Some 3rd party vendors will install an entry there to ensure the initialization and start-up of dependent services. E.g. Oracle places cluster services start-up entries here to ensure they start at boot time. It is important to review this file and verify all entries (in the `/etc/inittab`) are appropriate in your environment.

It is *highly* recommended to trap the Ctrl + Alt + Del key sequence in order to prevent accidental reboots. The following command uses the sed string processor to find the `ca::ctrlaltdel` string and add a `#` symbol to the front of it. This will comment-out the whole line – thereby disabling the entry:

```
sed -i 's/\(ca::ctrlaltdel:\)/#\1/g' /etc/inittab
```

It is generally recommended to make the default runlevel of a production system set to "3" - which means, multiuser *with* networking but *without* graphics. In other words – the X subsystem should not be running. It uses server compute cycles and is generally not needed. In this case, it should be removed – if not used. Setting the default runlevel is normally done via YaST at time of installation – or by using the *System Services (Runlevel)* tool in YaST. This tool will set the inittab for you, or you can make the modification manually (ensuring `id:3:initdefault` is in the inittab):

```
jupiter:~ # grep ':initdefault' /etc/inittab
id:3:initdefault:
```

To have changes in `/etc/inittab` become effective immediately, you can "tell" init to re-examine the inititab like this:

```
telinit q
```

Reviewing the System Services tool in YaST is a great way to determine what will be started on a system. The *System Services (Runlevel)* tool has a normal and *Expert* mode. Switch to *Expert* to see some of the boot services. These can be important for things like multipath, etc.

3.19 Restricting System Access from Servers and Networks

Normally a firewall is used to protect a server from other servers and networks – or to protect an environment from the server itself. However, you can also protect a server or more accurately, an individual service using a TCP wrapper (which is a function of `xinetd` – already discussed above).

The `xinetd` super server that comes with SUSE Linux Enterprise Server provides a built-in TCP wrapper functionality. This can be used to specifically define network services to accept or deny incoming connections from specified hosts and networks. The TCP wrappers implements access control through the use of two files, `/etc/hosts.allow` and `/etc/hosts.deny`. Note that the `hosts.allow` file takes precedence over the `hosts.deny` file. And you may want to change the permissions on the two configuration files since they are both world readable. An important difference between the security provided by a TCP wrapper vs. the use of netfilter (iptables) – is that netfilter works at the network layer (layer 2) and can provide security before the traffic goes up the stack to the application layers. The trade-off however is that a TCP wrapper allows for the use of a "banner" - or a welcome message (per service) and some other things. It is best to know what an organization requires – so the best recommendation can be made.

Generally, a best-practice strategy is to block all incoming requests by default, and allow only specific hosts or networks to connect. Commonly called - "deny-all, permit-few" this is the strategy that is proposed here. First, edit the `/etc/hosts.deny` file and add the following line:

```
ALL: ALL
```

Then, in order to accept incoming SSH connections from specific hosts (e.g. nodes `sles-ha1`, `sles-ha2` and `sles-ha3`, modify the `/etc/hosts.allow` file and add the following line:

```
sshd: sles-ha1 sles-ha2 sles-ha3
```

To accept incoming SSH connections from all servers from a specific network, add the name of the subnet to `/etc/hosts.allow`. Adjust the entry like this:

```
sshd: sles-ha1 sles-ha2 sles-ha3 .network.example.com
```

Remember, each service can be defined separately. In order to accept incoming portmap connections from the host at IP address `192.168.0.1` and from the subnet `192.168.5`, the following modification can be made to `/etc/hosts.allow`:

```
portmap: 192.168.0.1 192.168.5.
```

To accept connections from all servers on the subnet named .network.example.com but not from host badhost.network.example.com, add the following line to the `/etc/hosts.allow` file:

```
ALL: .network.example.com EXCEPT badhost.network.example.com
```

Here are other examples that show some features of TCP wrapper: If you just want to restrict ssh connections without configuring or using `/etc/hosts.deny`, you can add the following entries to `/etc/hosts.allow`:

```
sshd: rac1cluster rac2cluster rac3cluster
sshd: ALL: DENY
```

The version of TCP wrapper that comes with SUSE Linux Enterprise Server also supports the extended options documented in the `hosts_options` man page. Here is an example how an additional program can be spawned in e.g. the `/etc/hosts.allow` file:

```
sshd: ALL : spawn echo "Login from %c to %s" | mail -s "Login Info for %s"
 log@loghost
```

For information on the % expansions, see `man hosts_access`.

The TCP wrapper is quite flexible. And `xinetd` provides its own set of host-based and time-based access control functions. You can even tell `xinetd` to limit the rate of incoming connections. Various documentations about the `xinetd` super daemon on the Internet and should be considered recommended reading. Just remember the trade-offs between a TCP wrapper and the SuSEFirewall2 (netfilter/iptables).

3.20 Securing SSH

Many network services like telnet, rlogin, and rsh are vulnerable to eavesdropping which is one of several reasons why SSH should be used instead. The SUSE Linux Enterprise Server default configuration for SSH meets the security requirements for many environments. However, there are a few parameters in `/etc/ssh/sshd_config` that you may want to change.

The section on Restricting System Access from Servers and Networks shows how direct logins can be disabled for shared and system accounts including root. But it's prudent to disable direct root logins at the SSH level as well.

```
PermitRootLogin no
```

Also ensure to have privilege separation enabled where the daemon is split into two parts. With privilege separation a small part of the code runs as root and the rest of the code runs in a chroot jail environment.

```
UsePrivilegeSeparation yes
```

Since SSH protocol version 1 is not as secure you may want to limit the protocol to version 2 only:

```
Protocol 2
```

You may also want to prevent SSH from setting up TCP port and X11 forwarding if you don't need it:

```
AllowTcpForwarding no
X11Forwarding no
```

Ensure the StrictModes directive is enabled which checks file permissions and ownerships of some important files in the user's home directory like `~/.ssh`, `~/.ssh/authorized_keys` etc. If any checks fail, the user won't be able to log in.

```
StrictModes yes
```

Ensure that all host-based authentications are disabled. These methods should be avoided as primary authentication.

```
IgnoreRhosts yes
HostbasedAuthentication no
RhostsRSAAuthentication no
```

Disable sftp if it's not needed (by commenting it out with the `#`):

```
#Subsystem sftp /usr/lib/misc/sftp-server
```

After changing any directives make sure to restart the `sshd` daemon:

```
/etc/init.d/sshd restart
```

3.21 Securing Postfix

Postfix is a replacement for Sendmail and has several security advantages over Sendmail. Postfix is the default mail system in SUSE Linux Enterprise Server and consists of several small programs that each performs their own small task – most of these programs run in a chroot jail. This is another reason why Postfix is recommended over Sendmail.

Linux servers that are not dedicated mail or relay servers should not accept external e-mails. However, it is important for production servers to send local e-mails to a relay server – some security setups (e.g. the seccheck scripts) can be configured to send e-mails to someone other than `root`, even off the local system.

Verify the following parameters in `/etc/postfix/main.cf` are set to ensure that Postfix accepts only local e-mails for delivery (look towards the bottom of the file as the top portion is mostly commented-out example entries and explanations):

```
mydestination = $myhostname, localhost.$mydomain, localhost
inet_interfaces = localhost
```

The `mydestination` parameter lists all domains to receive e-mails for. The `inet_interfaces` parameter specifies the network to listen on. After reconfiguring Postfix, a restart of the mail system is necessary:

```
rcpostfix restart
```

Verify that Postfix is still listening for network requests (incoming) by running one of these commands from another host:

```
nmap -sT -p 25 remote_host
telnet <remote_host> 25
```

NOTE

Running these commands on the local host will provide inaccurate results since Postfix is supposed to accept connections from the local node – use an external host for correct results.

3.22 Filesystems: Securing NFS

NFS (Network File System) allows servers to share files over a network. But like all network services using NFS involves risks.

Here are some basic rules:

- NFS should not be enabled if not needed.
- If NFS is truly needed, use a TCP wrapper to restrict. remote access
- Ensure to export only to those hosts that really need access.
- Use a fully qualified domain name to diminish any spoofing attempts.
- Export only as read-only whenever possible.
- Use NFS only over TCP.

If you don't have shared directories to export, then ensure that the NFS service is *not* enabled nor running on the system:

Check the nfs service status:

```
jupiter:~ # /etc/init.d/nfsserver status
Checking for kernel based NFS server:
idmapd                                                    running
mountd                                                    unused
statd                                                     unused
nfsd                                                      unused
```

Check the current runlevels:

```
jupiter:~ # chkconfig -l nfsserver
nfsserver 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

3.22.1 Enabling and Starting NFS Server

If NFS must be used, it can be activated using the following commands on SUSE Linux Enterprise Server or more simply and securely with the YaST plug-in (ncurses). Access it directly from command line with `yast nfs-server` or `yast nfs-client` – or manually:

```
chkconfig nfs on
rcnfs start
```

Portmapper information:

```
jupiter:~ # rpcinfo -p server
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100003    2   udp   2049  nfs
```

```
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100005    1   udp    623  mountd
    100005    1   tcp    626  mountd
    100005    2   udp    623  mountd
    100005    2   tcp    626  mountd
    100005    3   udp    623  mountd
    100005    3   tcp    626  mountd
```

If you run it from an "untrusted" server or network, you should get the following output:

```
jupiter:~ # rpcinfo -p server
No remote programs registered.
```

3.22.2 Exporting NFS File Systems

To allow a client access to a file system or directory, the `/etc/exports` serves as the access control list. To give the network "network.example.com" read-only access to `/pub`, the entries in `/etc/exports` would look like as follows:

```
/pub *.network.example.com(ro,sync)
```

It is very important *not* to give write access to NFS clients if not absolutely needed! Entries in `/etc/exports` are exported read-only (`ro` option) by default. To allow servers `sles-ha1`, `sles-ha2` and `sles-ha3` read-write access to the `/data/MYSQL` directory, the entries in `/etc/exports` would look like as follows:

```
/data/MYSQL sles-ha1.example.com(rw,sync) sles-ha2.example.com(rw,sync)
 sles-ha3.example.com(rw,sync)
```

Note that the options *must not* be separated from the hostnames or networks with whitespace(s). Also, fully qualified domain names should *always* be used to diminish spoofing attempts. All entries in `/etc/exports` are exported with the `root_squash` option (“root squashing”) by default. This means that a root user on a client machine does not have root privileges (root access) to root-owned files on exported NFS file systems/directories. It is not recommended to turn “root squashing” off using the `no_root_squash` option! After you've made all your entries in `/etc/exports`, you can export all file systems/directories using the following command:

```
exportfs -a
```

To unexport all shared file systems/directories, run:

```
exportfs -ua
```

To see all shared file systems/directories, run:

```
jupiter:~ # showmount -e localhost
Export list for localhost:

/pub *.network.example.com/data/MYSQL
sles-ha1.example.com,sles-ha2.example.com,sles-ha3.example.com
```

3.22.3 Using NFS over TCP

If you need NFS, it is recommended to use NFS only over TCP since NFS over UDP is not secure. All 2.4 and later kernels support NFS over TCP on the client side. Server support for TCP appears in later 2.4 kernels and beyond.In order to mount a shared directory using NFS over TCP, it is necessary to use the `proto=tcp` mount option:

```
mount -o proto=tcp server_name:/pub /usr/local/pub
```

Verify that the target directory, in this case `/usr/local/pub`, exists on the client:

jupiter:~ # mount [...] *server_name*:/pub on /usr/local/pub type nfs (rw,proto=tcp,addr=192.168.1.110)

To have the shared directory mounted on the client at boot time, use the `/etc/fstab` file. For the above example, the `/etc/fstab` entry could look like this:

```
server_name:/pub /usr/local/pub nfs rsize=8192,wsize=8192,timeo=14,intr,tcp
 0 0
```

3.23 Copying Files Using SSH Without Providing Login Prompts

This example is needed in some cases in order to enable files to be copied over the network using ssh without providing an interactive login prompt. This allows trusted hosts to be set up – an example of federation.

SSH can allow a forced command using the “command” option. Using this option it is possible to disable scp (secure copy) and enforce every passed ssh command to be ignored. On the server side where you want to retrieve the file from, add the follow-

ing entry to the beginning of the SSH key in the ~/.ssh/authorized_keys2 file (the ~ represents a particular users "home" directory – root's home directory is /root – other users typically reside in /home/*username*):

```
command="/bin/cat ~/<file_name>" ssh-dss XXXYYYzzZ1122AAbbCC...{etc}
```

To copy now the file from the remote server, run the following command:

ssh *user@server local_file*

Since /bin/cat is run on the server side, its output has to be redirected to the local file.

Another approach is to replace the /bin/cat (referenced above) with your own script that checks the passed SSH commands by reading the environment variable $SSH_ORIGINAL_COMMAND. For example:

```
#!/bin/ksh
 if [[ $SSH_ORIGINAL_COMMAND = "File1" ||
       $SSH_ORIGINAL_COMMAND = "File2" ]]
 then
     /bin/cat $SSH_ORIGINAL_COMMAND
 else
     echo "Invalid file name!"
     exit 1
 fi
```

So you replace the /bin/cat portion with the script name in ~/.ssh/authorized_keys2, and run the following command to copy Foo1:

```
ssh user@server Foo1 > local_file
```

To copy Foo 2, run:

```
ssh user@server Foo2 > local_file
```

With the modifications above, every other variety of passed parameters will return errors.

3.24 Checking File Permissions and Ownership

The following sections deal with some of the ways in which the default permissions and file settings can be modified to enhance the security of a host. It is important to note

that the use of the default SUSE Linux Enterprise Server utilities like `seccheck` - can be run to lock down and improve the general file security and user environment. However, it is beneficial to understand how to modify these things.

SUSE Linux Enterprise Server hosts include 3 defaults settings for file permissions: `permissions.easy`, `permissions.secure`, and `permissions.paranoid`, all located in the `/etc` directory. The purpose of these files is to define special permissions, such as world-writable directories or, for files, the setuser ID bit (programs with the setuser ID bit set do not run with the permissions of the user that has launched it, but with the permissions of the file owner, in most cases `root`).

Administrators can use the file `/etc/permissions.local` to add their own settings. The easiest way to implement one of the default permission rule-sets above is to use the *Local Security* module in YaST.

Each of the following topics will be modified by a selected rule-set, but the information is important to understand on its own.

3.25 Default umask

The `umask` (user file-creation mode mask) command is a shell built-in command which determines the default file permissions for newly created files. This can be overwritten by system calls but many programs and utilities make use of `umask`. By default, SUSE sets `umask` to `022`. You can modify this by changing the value in `/etc/profile`.

The `id` command will print out the current user identity information. Example from a non-root prompt:

```
jupiter:~ > id
uid=1000(ne0) gid=100(users) groups=16(dialout),33(video),100(users)
```

And to determine the active umask – use the umask command:

```
jupiter:~ > umask
0022
```

Now for comparison sake – the same commands from the root user:

```
jupiter:~ # id
uid=0(root) gid=0(root) groups=0(root)
jupiter:~ # umask
```

```
0022
```

3.26 SUID/SGID Files

When the SUID (set user ID) or SGID (set group ID) bits are set on an executable, it executes with the UID or GID of the owner of the executable rather than that of the person executing it. This means that, for example, all executables that have the SUID bit set and are owned by root are executed with the UID of root. A good example is the passwd command that allows ordinary users to update the password field in the /etc/shadow file which is owned by root.

But SUID/SGID bits can be misused when the SUID/SGID executable has a security hole. Therefore, you might want to search the entire system for SUID/SGID executables and document it. For example, ensure that code developers don't set SUID/SGID bits on their programs if it's not an absolute requirement. Very often you can use workarounds like removing just the executable bit for world/others. However, a better approach is to change the design of the software if possible.

To search the entire system for SUID or SGID files, you can run the following command:

```
find / -path /proc -prune -o -type f -perm +6000 -ls
```

The `-prune` option in this example is used to skip the /proc file system.

3.27 World-Writable Files

World-writable files are a security risk since it allows anyone to modify them. Additionally, world-writable directories allow anyone to add or delete files. To locate world-writable files and directories, you can use the following command:

```
find / -path /proc -prune -o -perm -2 ! -type l -ls
```

The `! -type l` parameter skips all symbolic links since symbolic links are always world-writable. However, this is not a problem as long as the target of the link is not world-writable, which is checked by the above find command.

World-Writable directories with sticky bit such as the `/tmp` directory do not allow anyone except the owner of a file to delete or modify it in this directory. The sticky bit makes files stick to the user who created it and it prevents other users from deleting and renaming the files. Therefore, depending on the purpose of the directory world-

writable directories with sticky are usually not an issue. An example is the /tmp directory:

```
jupiter:~ > ls -ld /tmp
drwxrwxrwt 18 root root 16384 Dec 23 22:20 /tmp
```

The t mode, which denotes the sticky bit, allows files to be deleted and renamed only if the user is the owner of this file or the owner of the directory.

3.28 Orphaned or Unowned Files

Files not owned by any user or group might not necessarily be a security problem in itself. However, unowned files could pose a security problem in the future. For example, if a new user is created and the new users happens to get the same UID as the unowned files have, then this new user will automatically become the owner of these files.

To locate files not owned by any user or group, use the following command:

```
find / -path /proc -prune -o -nouser -o -nogroup
```

3.29 Various Account Checks

3.29.1 Unlocked Accounts

It is important that all system and vendor accounts that are not used for logins are locked.To get a list of unlocked accounts on your system, you can check for accounts that do *not* have an encrypted password string starting with ! or * in the /etc/shadow file. If you lock an account using passwd -l, it will put a !! in front of the encrypted password, effectively disabling the password. If you lock an account using usermod -L, it will put a ! in front of the encrypted password. Many system and shared accounts are usually locked by default by having a * or !! in the password field which renders the encrypted password into an invalid string. Hence, to get a list of all unlocked (encryptable) accounts, run (egrep is used to allow use of regular-expressions):

```
egrep -v '.*:\*|:\!' /etc/shadow | awk -F: '{print $1}'
```

Also make sure all accounts have a x in the password field in /etc/passwd. The following command lists all accounts that do not have a x in the password field:

```
grep -v ':x:' /etc/passwd
```

A `x` in the password fields means that the password has been shadowed, i.e. the encrypted password has to be looked up in the `/etc/shadow` file. If the password field in `/etc/passwd` is empty, then the system will not lookup the shadow file and it will not prompt the user for a password at the login prompt.

3.29.2 Unused Accounts

All system or vendor accounts that are not being used by users, applications, by the system or by daemons should be removed from the system. You can use the following command to find out if there are any files owned by a specific account:

```
find / -path /proc -prune -o -user account -ls
```

The `-prune` option in this example is used to skip the /proc file system. If you are sure that an account can be deleted, you can remove the account using the following command:

```
userdel -r account
```

Without the `-r` option userdel will not delete the user's home directory and mail spool (`/var/spool/mail/user`). Note that many system accounts have no home directory.

3.30 Single User Mode Password for `root`

SUSE includes the following entry in the `/etc/inittab` file to ensure that a root password is required for Single User Mode logins:

```
~~:S:respawn:/sbin/sulogin
```

NOTE

This works well and will even restart the sulogin if terminated, but it can be easily circumvented!

The GRUB prompt can accept parameters allowing the execution of an alternate program – like the Bash shell (e.g. `init=/bin/bash`). This will place you at a root

shell prompt without a password. This further enhances the need to password protect the GRUB boot loader.

3.31 Enabling Password Aging

Password expirations are a general best practice—but might need to be excluded for some system and shared accounts (e.g. Oracle, etc). Expiring password on those accounts could lead to system outages if the application account expires.

Typically a corporate policy should be developed that dictates rules/procedures regarding password changes for system and shared accounts. However, normal user account passwords should expire automatically. The following example shows how password expiration can be setup for individual user accounts.

The following files and parameters in the table can be used when a new account is created with the `useradd` command. Settings such as these are stored for each user account in the `/etc/shadow` file. If using the YaST tool (*User and Group Management*) to add users, the settings are available on a per-user basis. Here are the various settings—some of which can also be system-wide (e.g. modification of `/etc/login.defs` and `/etc/default/useradd`):

`/etc/login.defs`	`PASS_MAX_DAYS`	Maximum number of days a password is valid.
`/etc/login.defs`	`PASS_MIN_DAYS`	Minimum number of days before a user can change the password since the last change.
`/etc/login.defs`	`PASS_WARN_AGE`	Number of days when the password change reminder starts.
`/etc/default/useradd`	`INACTIVE`	Number of days after password expiration that account is disabled.

/etc/default/useradd	EXPIRE	Account expiration date in the format YYYY-MM-DD.

NOTE

Users created prior to these modifications will not be affected.

Ensure that the above parameters are changed in the /etc/login.defs and /etc/default/useradd files. Review of the /etc/shadow file will show how these settings get stored after adding a user.

To create a new user account, execute the following command:

```
useradd -c "Test User" -g userstest
```

The -g option specifies the primary group for this account:

```
jupiter:~ # id test
uid=509(test) gid=100(users) groups=100(users)
```

The settings in /etc/login.defs and /etc/default/useradd are recorded for the test user in the /etc/shadow file as follows:

```
jupiter:~ # grep test /etc/shadow
test:!!:12742:7:60:7:14::
```

Password aging can be modified at any time by use of the chage command. To disable password aging for system and shared accounts, you can run the following chage command:

```
chage -M 99999 system_account_name
```

To get password expiration information:

```
chage -l system_account_name
```

For example:

```
jupiter:~ # chage -l test
Minimum: 7
Maximum: 60
Warning: 7
Inactive: 14
Last Change: Jan 11, 2011
Password Expires: Mar 12, 2011
Password Inactive: Mar 26, 2011
```

```
Account Expires: Never
```

3.32 Stronger Password Enforcement

On an audited system it is important to restrict people from using simple passwords that can be cracked too easily. However, if the passwords being enforced are too strong, people will write them down. Strong passwords that are written down are not much safer than weak passwords. Some will argue that strong passwords protect you against dictionary attacks and those type of attacks can be defeated by locking accounts after a few failed attempts. However, this is not always an option. If set up like this, locking system accounts could bring down your applications and systems which would be nothing short of a denial of service attack – another issue.

At any rate, it is important to practice effective password management safety. Most companies require that passwords have at the very least a number, another alpha character, and one upper case letter. Policies vary – but maintaining a balance between password strength/complexity and management is sometimes difficult.

3.33 Leveraging an Effective pam-stack

Linux-PAM (Pluggable Authentication Modules for Linux) is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users.

It is strongly recommended to familiarize oneself with the capabilities of PAM – and how this architecture can be leveraged to provide the “best” authentication setup for an environment. This configuration can be done once – and implemented across all systems (a standard) or can be enhanced for individual hosts (enhanced security – by host / service / application). The key is to realize how flexible the architecture is and how incredibly flexible it is.

To learn more about the PAM architecture – you can find PAM documentation on a SUSE Linux Enterprise Server system in the `/usr/share/doc/packages/pam` directory (in a variety of formats).

The following discussions are examples of how to modify the default pam stacks—specifically around password policies—e.g. password strength, password re-use and account locking. While these are only a few of the possibilities, they serve as a good start and demonstrate PAM's flexibility.

3.33.1 Password Strength

SUSE Linux Enterprise Server can leverage the `pam_cracklib` library to test for weak passwords – and to suggest using a stronger one if it determines obvious weakness. The following parameters represent an example or policy that could be part of a corporate password policy or something required due to audit constraints.

The pam libraries follow a defined “flow”—and in most cases, the best way to design the perfect stack is to consider all of the requirements and policies—and draw out a flow chart.

Table 3.2: *Sample rules/constraints for password enforcement*

`pam_cracklib.so`	`minlen=8`	Minimum length of password is 8
`pam_cracklib.so`	`lcredit=-1`	Minimum number of lower case letters is 1
`pam_cracklib.so`	`ucredit=-1`	Minimum number of upper case letters is 1
`pam_cracklib.so`	`dcredit=-1`	Minimum number of digits is 1
`pam_cracklib.so`	`ocredit=-1`	Minimum number of other characters is 1

To setup these password restrictions, use the `pam-config` tool and specify the parameters you want to configure. For example, the minimum length parameter could be modified like this:

```
pam-config -a --cracklib-minlen=8 --cracklib-retry=3 \
--cracklib-lcredit=-1 --cracklib-ucredit=-1 --cracklib-dcredit=-1 \
--cracklib-ocredit=-1 --cracklib
```

Now verify that the new password restrictions work for new passwords. Simply login to a non-root account and change the password using the `passwd` command. Note that the above requirements are not enforced if you run the `passwd` command under root.

3.33.2 Restricting Use of Previous Passwords

The `pam_unix2` module parameter remember can be used to configure the number of previous passwords that cannot be reused. And the `pam_cracklib` module parameter `difok` can be used to specify the number of characters that must be different between the old and the new password.

The following example describes how to implement password restrictions on a system so that a password cannot be reused for at least 6 months and that at least 3 characters must be different between the old and new password.

Recall that in the section Section 3.31, "Enabling Password Aging" (page 53) we set `PASS_MIN_DAYS` to `7`, which specifies the minimum number of days allowed between password changes. Therefore, if `pam_unix2` is configured to remember `26` passwords, then the previously used passwords cannot be reused for at least 6 months (26*7 days).

Here is an example of an enhanced pam stack. It is possible to edit the `/etc/pam.d/common-auth` file to add/change modules used and how they react. Consider the following `pam_cracklib` and `pam_unix2` arguments—keeping in mind how the pam rules are processed:

```
auth        required      pam_env.so
auth        sufficient    pam_unix2.so likeauth nullok
auth        required      pam_deny.so
account     required      pam_unix2.so
account     sufficient    pam_succeed_if.so uid < 100 quiet
account     required      pam_permit.so
password    requisite     pam_cracklib.so retry=3 minlen=8 lcredit=-1
                             ucredit=-1 dcredit=-1 ocredit=-1 difok=3
password    sufficient    pam_unix2.so nullok use_authtok md5 shadow
                             remember=26
password    required      pam_deny.so
session     required      pam_limits.so
session     required      pam_unix2.so
```

NOTE: `/etc/security/opasswd`

If the /etc/security/opasswd doesn't exist, create the file.

```
jupiter:~ # ls -l /etc/security/opasswd
-rw------- 1 root root 0 Dec 8 06:54 /etc/security/opasswd
```

3.33.3 Locking User Accounts After Too Many Login Failures

It is not generally recommend that a host automatically locks system and shared accounts after too many failed login or su attempts. This could lead to outages if the application's account gets locked due to too many login failures like in this example for an oracle shared account:

```
jupiter:~ # su oracle -c id
su: incorrect password
```

This could be an easy target for a denial of service attack. The following example shows how to lock only individual user accounts after too many failed su or login attempts. Add the following two lines to the `/etc/pam.d/common-auth`:

```
auth      required      pam_tally.so onerr=fail no_magic_root
[...]
auth      required     pam_tally.so per_user deny=5 no_magic_root reset
```

The first added line counts failed login and failed su attempts for each user. The default location for attempted accesses is recorded in `/var/log/faillog`.

The second added line specifies to lock accounts automatically after `5` failed login or su attempts (`deny=5`). The counter will be reset to 0 (reset) on successful entry if `deny=`*`n`* was not exceeded. But you don't want system or shared accounts to be locked after too many login failures (denial of service attack).

It is also possible to add the `lock_time=`*`n`* parameter, and then optionally the `unlock_time=`*`n`* parameter. For example, setting the `lock_time=60` would deny access for 60 seconds after a failed attempt. The `unlock_time=`*`n`* option would then allow access after n seconds after an account has been locked. If this option is used the user will be locked out for the specified amount of time after he exceeded his maximum allowed attempts. Otherwise the account is locked until the lock is removed by a manual intervention of the system administrator. See the `pam_tally` man page for more information.

To exempt system and shared accounts from the `deny=`*`n`* parameter, the per_user parameter was added to the module. The per_user parameter instructs the module *not* to

use the `deny=n` limit for accounts where the maximum number of login failures is set explicitly. For example:

```
jupiter:~ # faillog -u oracle -m -1
    Username   Failures  Maximum  Latest
    oracle     0         -1       Fri Dec 10 23:57:55 -0600 2005 on unknown
```

The faillog command with the option `-m -1` has the effect of not placing a limit on the number of failed logins—effectively disabling the option. To instruct the module to activate the `deny=n` limit for this account again, run:

```
faillog -u oracle -m 0
```

By default, the maximum number of login failures for each account is set to zero (0) which instructs pam_tally to leverage the `deny=n` parameter. To see failed login attempts, run:

```
faillog
```

To unlock a locked account (after too many login failures), use the `-r` option:

```
faillog -u user -r
```

Make sure to test these changes (and *any* changes – for that matter) thoroughly on your system using ssh and su, and make sure the `root` id does not get locked! To lock/unlock accounts manually, you can run one of the following commands:

Locking

```
passwd -l user
         usermod -L user
```

Unlocking

```
passwd -u user
usermod -U user
```

3.34 Preventing Accidental Denial of Service

Linux allows you to set limits on the amount of system resources that users and groups can consume. This is also very handy if bugs in programs cause them to use up too much resources (e.g. memory leaks), slow down the machine, or even render the system unusable. Incorrect settings can allow programs to use too many resources which may make the server unresponsive to new connections or even local logins (e.g. if a

program uses up all available file handles on the host). This can also be a security concern if someone is allowed to consume all system resources and therefore cause a denial of service attack – either unplanned or worse, planned. Setting resource limits for users and groups may be an effective way to protect systems, depending on the environment.

3.34.1 Example for Restricting System Resources

The following example demonstrates the practical usage of setting or restricting system resource consumption for an Oracle user account. For a list of system resource settings, see `/etc/security/limits.conf` or `man limits.conf`.

Most shells like Bash provide control over various resources (e.g. the maximum allowable number of open file descriptors or the maximum number of processes) that are available on a per/user basis. To examine all current limits in the shell execute:

```
ulimit -a
```

For more information on `ulimit` for the Bash shell, examine the Bash man pages.

IMPORTANT: Setting Limits for SSH Sessions

Setting "hard" and "soft" limits might not behave as expected when using an SSH session. To see valid behavior it may be necessary to login as root and then su to the id with limits (e.g. oracle in these examples). Resource limits should also work assuming the application has been started automatically during the boot process. It may be necessary to set UsePrivilegeSeparation in /etc/ssh/sshd_config to "no" and restart the SSH daemon (rcsshd restart) if it seems that the changes to resource limits are not working (via SSH). However this is not generally recommended – as it weakens a systems security.

In this example, a change to the number of file handles or open files that the Oracle user can use is made by editing `/etc/security/limits.conf` as `root` making the following changes:

```
oracle           soft    nofile          4096
oracle           hard    nofile          63536
```

The “soft” limit in the first line defines the number of file handles (open files) that the `oracle` user will have after login. If the user sees error messages about running out

of file handles, then the user can increase the number of file handles like in this example up to the “hard” limit (in this example 63536) by executing:

```
ulimit -n 63536
```

You can set the “soft” and “hard” limits higher if necessary.

NOTE

It is important to be judicious with the usage of ulimits – allowing a "hard" limit for “nofile” for a user that equals the kernel limit (`/proc/sys/fs/file-max`) is very bad! If the user consumes all the available file handles – the system will not be able to initiate new logins as accessing (opening) PAM modules which are required for performing a login will not be possible.

You also need to ensure that `pam_limits` is configured in `/etc/pam.d/common-auth`, or in an individual service (like SSH, su, login, telnet, etc.) config:

`/etc/pam.d/sshd` (for SSH)
`/etc/pam.d/su` (for su)
`/etc/pam.d/login` (local logins and telnet)

If you don't want to enable it for all logins, there is a specific PAM module that will read the `/etc/security/limits.conf` file. Entries in pam configuration directives will have entries like:

```
session     required      /lib/security/pam_limits.so
session     required      /lib/security/pam_unix.so
```

It is important to note that changes are not immediate and require a new login session:

```
jupiter:~ # su - oracle
jupiter:~ > ulimit -n
4096
```

Note that these examples are specific to the Bash shell - `ulimit` options are different for other shells. The default limit for the oracle account is now `4096` and that the oracle user can increase the number of file handles up to `63536` (based on the settings enacted):

```
jupiter:~ # su - oracle
jupiter:~ > ulimit -n
4096
jupiter:~ > ulimit -n 63536
jupiter:~ > ulimit -n
63536
```

Making this permanent requires the addition of the setting, `ulimit -n 63536`, (again, for Bash) to the users profile (`~/.bashrc` or `~/.profile` file) which is the user start-up file for the Bash shell on SUSE Linux (to verify your shell run: echo `$SHELL`). To do this you could simply type (or copy/paste – if you are reading this on the system) the following commands for the oracle user's Bash shell:

```
jupiter:~ # su - oracle
jupiter:~ > cat >> ~oracle/.bash_profile << EOF
ulimit -n 63536
EOF
```

3.35 Displaying Login Banners

In many cases (and after per corporate policy) it is necessary to place a banner on login screens on all servers for legal or audit policy reasons (and to potentially deter intruders) among other things.

If you want to print a legal banner after a user logs in using ssh, local console etc., you can leverage the `/etc/motd` (motd = message of the day) file. The file exists on SUSE – however it is empty. Simply add content to the file that is applicable/required by the organization.

NOTE: Banner Length

Try to keep the content to a single page (or less) – as it will scroll the screen if it doesn't fit.

For SSH you can edit the "Banner" parameter in the `/etc/ssh/sshd_config` file which will then appropriately display the banner text before the login prompt. For local console logins you can edit the `/etc/issue` file which will display the banner before the login prompt.For GDM you could make the following changes to require a user to acknowledge the legal banner by selecting 'Yes' or 'No'. Edit the `/etc/gdm/PreSession/Default` file and add the following lines at the beginning of the script:

```
if ! gdialog --yesno '\nThis system is classified...\n' 10 10; then
    sleep 10
    exit 1;
fi
```

Obviously the *`This system is classified...`* test should be replaced with the valid text – and it is important to note that this dialog will not prevent a login to progress. The existence of a "Cancel" button is merely incidental.

3.36 Miscellaneous

3.36.1 Host-Based Linux Monitoring and Intrusion Detection

Before you place a host into production or even on a network, consider the use of an system integrity checker – like `seccheck` (already discussed in Section 3.4, “Verifying Security Action with seccheck” (page 18))—so in the event of unauthorized changes, notifications can happen. Also consider the use of an intrusion detection environment, like AIDE – the Advanced Intrusion Detection Environment.

AIDE is a GPL licensed and open source intrusion detection system. It could be considered a system “fingerprinting” mechanism. AIDE works by creating a database containing information about the files on your system. The database is created from rules laid out in the configuration file aide.conf. When AIDE is run, this database is referenced to check for changes (or created for the first time). Assuming a comparison check is being run, any changes not permitted by the configuration file are reported.

By leveraging AIDE—storing a copy of the host's database in a secure location—and comparing it (on a scheduled basis or as part of a forensic effort), system integrity/insurance can be a matter of heuristics and procedure. If an intruder compromises your system you the comparison effort will enable an administrator or security officer to know what has changed on the host. The initial database should be created as a final step – *before* a system gets deployed into production.

It is outside the scope of this article to cover Linux Monitoring and detailed Intrusion Detection systems (IDS) or solutions – however there is a plethora of information of configuring AIDE or other solutions and many informative articles on the web.

3.36.2 Connect Accounting Utilities

Here is a list of commands you can use to get data about user logins:

`who` Shows a listing of currently logged-in users.

`w` Shows who is logged on and what they are doing.

last Shows a list of last logged-in users, including login time, logout time, login IP address, etc.

lastb Same as `last`, except that by default it shows a log of the file `/var/log/btmp`, which contains all the bad login attempts.

lastlog This command reports data maintained in `/var/log/lastlog`, which is a record of the last time a user logged in.

ac Prints out the connect time in hours on a per-user basis or daily basis etc. This command reads `/var/log/wtmp`.

dump-utmp Converts raw data from /var/run/utmp or `/var/log/wtmp` into ASCII-parsable format.

Also check the `/var/log/messages` file.

3.36.3 Other

Finally, the following couple items might not be (specifically) security related, but misconfiguration can cause many problems – and should be reviewed:

- Resolver (`/etc/hosts`, `/etc/resolv.conf`, `/etc/nsswitch.conf`).
- NTP configuration (`/etc/ntp.conf`).